Automatic Synthesis and Fault-Tolerant Experiments on an Evolvable Hardware Platform[†1]

Adrian Stoica, Didier Keymeulen, V. Duong and C. Salazar-Lazaro
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109
818-354-2190
adrian.stoica@jpl.nasa.gov

Abstract— Outer solar system exploration and missions to comets and planets with severe environmental conditions require long-term survivability of space systems. This challenge has recently been approached with new ideas, such as using mechanisms for hardware adaptation inspired from biology. The application of evolution-inspired formalisms to hardware design and self-configuration lead to the concept of evolvable hardware (EHW). EHW refers to self-reconfiguration of electronic hardware by evolutionary/genetic reconfiguration mechanisms. The paper describes a fine-grained Field Programmable Transistor Array (FPTA) architecture for reconfigurable hardware, and its implementation on a VLSI chip. A first experiment illustrates automatic synthesis of electronic circuits through evolutionary design with the chip-in-the-loop. The chip is rapidly reconfigured to evaluate candidate circuit designs. A second, fault-tolerance experiment shows how evolutionary algorithms can recover functionality after being subjected to faults, by finding new circuit configurations that circumvent the faults.

TABLE OF CONTENTS

1. INTRODUCTION

Long-term survivability of space systems, as required, for example, by outer solar system exploration and missions to comets and planets with severe environmental conditions, has recently been approached with new ideas, such as the use of biology-inspired mechanisms for hardware adaptation. The application of evolution-inspired formalisms to hardware design and self-configuration lead to the concept of evolvable hardware (EHW). In the narrow sense EHW refers to self-reconfiguration of electronic hardware by evolutionary/genetic reconfiguration mechanisms. In a broader sense, EHW refers to various forms of hardware, from sensors and antennas to complete evolvable space systems that could adapt to changing environments and, moreover, increase their performance during the mission.

There are two main benefits EHW can bring to spacecraft survivability. Firstly, EHW can help preserving existing functions, in conditions where hardware is subject to faults, aging, temperature drifts and radiation, etc. Secondly, new functions can be generated (more precisely new hardware configurations can be synthesized to provide required functionality) when needed.

This paper reports on experiments that illustrate how evolutionary algorithms can design analog and digital circuits and recover functionality when lost due to faults, by finding new circuit configurations that circumvent the faults. The search for an electronic circuit realization of a desired transfer characteristic can be made in software as in *extrinsic* evolution, or in hardware as in *intrinsic* evolution. In extrinsic evolution the final solution is downloaded to (or becomes a blueprint for) the hardware. In *intrinsic* evolution the hardware actively participates in the circuit evolutionary process and is the support on which candidate solutions are evaluated.

A variety of circuits have been synthesized through evolutionary means. For example, Koza used Genetic Programming (GP) to grow an "embryonic" circuit to one that satisfies desired requirements [1]. This approach was used for evolving a variety of circuits, including filters and computational circuits. An alternative encoding technique for analog circuit synthesis, which has the advantage of reduced computational load was used by Lohn and Colombano[2] for automated filter design. On-chip evolution was demonstrated by Thompson [3] using an Field Programmable Gate Array (FPGA) as the programmable device, and a Genetic Algorithm (GA) as

[†] In Proceedings of the IEEE Aerospace Conference, March 18-25, 2000, Big Sky, MT, USA.

the evolutionary mechanism. More details on current work in evolvable hardware are found in [4-7]. Evolutions of analog circuits reported in [1] and [2] were performed in simulations without concern for a physical implementation. It shows that evolution can lead to circuit designs that compete, or even exceed in performance those of humans. Current programmable analog devices are very limited in capabilities and do not support the implementation of the resulted design (but, in principle, one can test their validity in circuits built from discrete components, or in an ASIC (Application Specific Integrated Circuit)). More recently, evolutionary experiments were performed on Field Programmable Analog Arrays [18] and ASIC [11].

There is another characteristic that makes electronic devices an attractive domain for applying evolution; the possibility to produce electronic systems that are inherently insensitive to faults such as silicon defects by using evolution in hardware to design fault-tolerant or highly reliable systems. The evolution is even able to self-repair on-line by exploiting defective components as if they were working parts [15-16].

This paper is organized as follows: Section 2 presents an evolution-oriented architecture for reconfigurable hardware based on the concept of Field Programmable Transistor Array. Section 3 presents the experimental setup, including details of the evolutionary design tool, the FPTA chip and the hardware evaluation board. Section 4 presents automatic synthesis of an electronic circuit by intrinsic evolution (on FPTA chips). Section 5 describes a fault-tolerant experiment in which functionality is recovered after a fault. Section 6 presents some lessons learned from the experiments and section 7 concludes the paper.

2. TOWARD EVOLUTION-ORIENTED CHIPS

In the context of electronic synthesis on reconfigurable devices, the architectural configurations are encoded in "chromosomes" that define the state of the switches connecting elements in the reconfigurable hardware. The main steps in evolutionary synthesis of electronic circuits are the following. First, a population of chromosomes is randomly generated to represent a pool of circuit architectures. The chromosomes are converted into circuit models (for extrinsic EHW) or control bitstrings downloaded to programmable hardware (intrinsic EHW). Circuit responses are compared against specifications of a target response and individuals are ranked based on how close they come to satisfying it. Preparation for a new iteration loop involves generation of a new population of individuals from the pool of the best individuals in the previous generation. Here, some individuals are taken as they were and some are modified by genetic operators, such as chromosome crossover and mutation. The process is repeated for a number of generations, resulting in increasingly better individuals. The process is usually ended after a given number of generations, or when the closeness to the target response has been reached. In practice, one or several solutions may be found among the individuals of the last generation.

Current efforts in the evolution of hardware have been limited to simple circuits [5]. For experiments with digital circuits, this limitation may be caused by a lack of power of evolutionary techniques in such search spaces. For analog circuits the limitation appears to come from a lack of appropriate reconfigurable analog devices to support the search. This precludes searches directly in hardware and requires evolving on hardware models. Such models require evaluation with circuit simulators such as SPICE; the simulators need to solve differential equations and, for anything beyond simple circuits, they require too much time for practical searches of millions of circuit solutions. A hardware implementation offers a big advantage in evaluation time for a circuit; the time for evaluation is determined by the goal function. For example, considering an A/D converter operating at a 100 kHz sampling rate the electronic response of the A/D converter is available within 10 microseconds, compared to (an over-optimistic) 1 second on a fast computer running SPICE; this advantage increases with the complexity of the circuits. In this case the 10^5 speedup would allow evaluations of populations of millions of individuals in seconds instead of days.

Most reconfigurable devices are digital, and while several levels of granularity are in use, the most common ones are configurable at the gate-level. In the analog programmable devices the reconfigurable active elements are Operational Amplifiers, such as in Field Programmable Analog Arrays (FPAA) with only very coarse granularity and few programmable components, allowing specified functionality with good precision, having a limited range of possible EHW experiments. The optimal choice of elementary block type and granularity is task dependent. At least for experimental work in evolvable hardware, it appears a good choice to build reconfigurable hardware based on elements of the lowest level of granularity. Virtual higher-level building blocks can be considered by imposing programming constraints. An example of this would entail forcing groups of elementary cells to act as a whole (e.g. certain parts of their configuration bitstrings with the interconnections for the N transistors implementing a NAND would be frozen). Ideally, the “virtual blocks” for evolution should be automatically defined/clustered during evolution (an equivalent of the Automatically Defined Functions predicted and observed in software evolution).

The idea of a field programmable transistor array was introduced first in [11]. The FPTA is a concept design for hardware reconfigurable at transistor level. As both analog and digital CMOS circuits ultimately rely on functions implemented with transistors, the FPTA appears as a versatile platform for the synthesis of both analog and

digital (and mixed-signal) circuits. Further, it is considered a more suitable platform for synthesis of analog circuitry than existing FPGAs or FPAAs, extending the work on evolving simulated circuits to evolving analog circuits directly on the chip. The FPTA module is an array of transistors interconnected by programmable switches. The status of the switches (ON or OFF) determines a circuit topology and consequently a specific response.

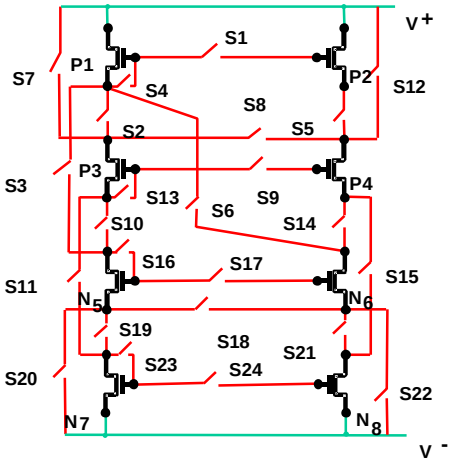

Figure 1 Module of the Programmable Transistor Array

Thus the topology can be considered as a function of switch states, and can be represented by a binary sequence, such as "1011…", where by convention one can assign 1 to a switch turned ON and 0 to a switch turned OFF. The FPTA architecture allows the implementation of bigger circuits by cascading FPTA modules with external wires.

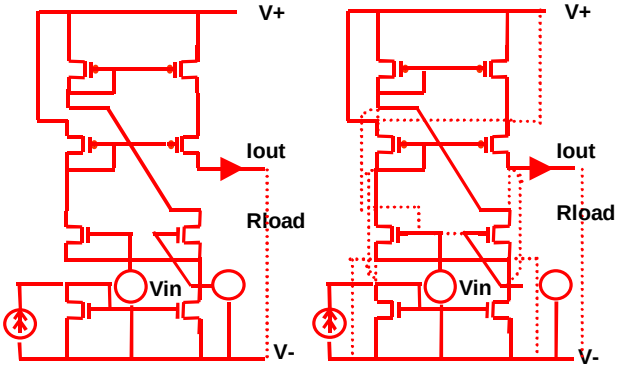

Figure 2 Schematic of a simple circuit implemented on a FPTA module (with leakage through the finite resistance of OFF switches as dotted lines on the right figure).

To offer sufficient flexibility the module has all transistor terminals connected via switches to expansion terminals (except those connected to power and ground). Issues related to chip expandability were treated in [11]. Figure 1 illustrates an example of a FPTA module consisting of 8 transistors and 24 programmable switches. In this example the transistors P1-P4 are PMOS and N5-N8 are NMOS, and the switch-based connections are in sufficient number to allow a majority of meaningful topologies for the given transistor arrangement, and yet less than the total number of possible connections. Programming the switches ON and OFF defines a circuit for which the effects of non-zero, finite impedance of the switches can be neglected in the first approximation. An example of a circuit drawn with this simplification is given in Figure 2.

3. Test Bed for Evolutionary Experiments

An evolutionary design tool was developed to facilitate experiments in simulated and hardware evolution [17]. The tool illustrated in Figure 3 can be used for synthesis and optimization of new devices, circuits, or architectures for reconfigurable hardware. The tool proved very useful in testing architectures of reconfigurable HW and demonstrating evolution on a dedicated reconfigurable chip. In its current implementation the tool uses the public domain Parallel Genetic Algorithm package, PGAPack, a public domain version of SPICE 3F5 as circuit simulator and an evolvable hardware test bed built around LabView. An interface code links the GA with the simulator and with the hardware where potential designs are evaluated, while a GUI allows easy problem formulation and visualization of results. At each generation the GA produces a new population of binary chromosomes, which get converted into voltages in Netlists that describe candidate circuit designs and into configuration bits for the reconfigurable devices. Netlists are further simulated by SPICE and configuration bits are downloaded into the hardware device by LabView. More details about the tool are given in [10],[17].

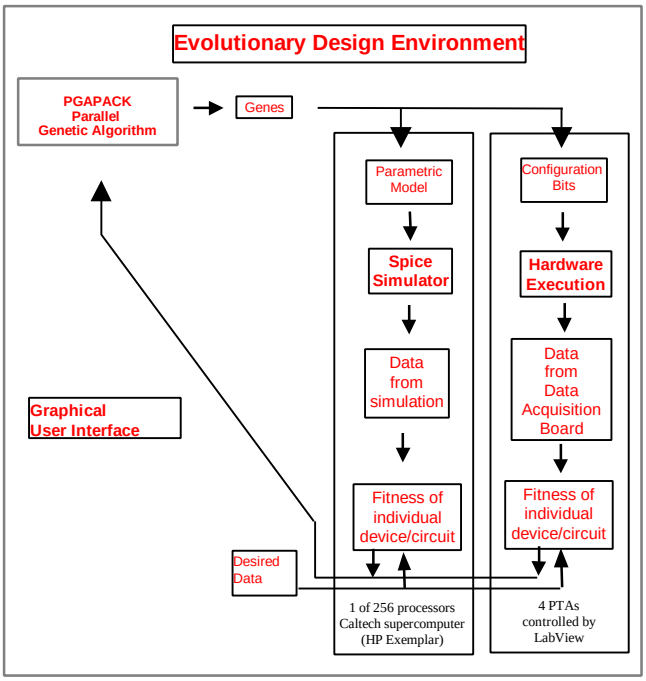

Figure 3 Environment for evolutionary design.

After successful evolution on the simulated FPTA a test chip implementing the FPTA architecture was developed. Circuit evolutionary synthesis directly on the chip became possible at an expected accelerated pace of over two orders of magnitude compared to the simulation on the supercomputer (estimated ~5 seconds compared to ~20 minutes for the experiment described). In the experimental simulations, the size of the transistors was fixed. The programmable switches were implemented with transistors, acting as simple T-gate switches.

Each chip contains one FPTA module and was fabricated as a Tiny Chip through MOSIS, using 0.5-micron CMOS technology. The test board with four chips mounted on it is illustrated in Figure 4.

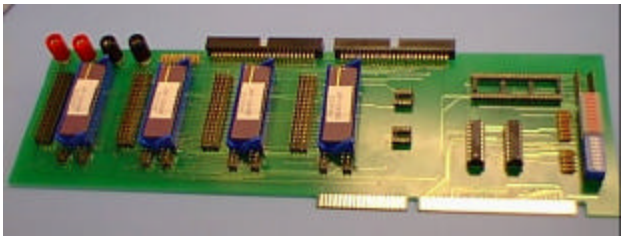

Figure 4 A test board with four FPTA chips

The hardware evaluation board is controlled by National Instruments data acquisition hardware and software (LabView) and integrated into the evolutionary design envionment.

4. Automatic Synthesis of a New Function

The following experiment performed in hardware on the FTPA chip illustrates the evolutionary synthesis of to computational circuit. The desired functionality is a nonlinear DC input-output characteristic (a Gaussian current-voltage characteristic). Four chips were programmed in parallel with bit-string configurations corresponding to four individuals of a population of 1000; after evaluation the chips were reprogrammed with the chromosome of the next four individuals, and so on until all 1000 in one generation were tested. Evolution led to "Gaussian" circuit solutions within 20-30 generations. The current speed of evaluation is 1000 circuits in 8.25 seconds using the four FPTA chips in parallel; another order of magnitude speed-up is expected when some existing data acquisition bottlenecks will be solved.

The following GA parameters were used:
Population: 1000, Chromosome size: 24 bits for 1 FPTA, and 52 to 88 bits for 2 FPTAs (the number depends on interconnection schemes), Evaluation samples: 30, Mutation rate: 4%, Crossover rate: 70%, Tournament Selection: 20 individuals, Elite Strategy: 9% population size (88 individuals), Fitness Function: Square Root Mean Error.

The response of four mutants is illustrated in the screen capture shown in Figure 5 (LabView display of the signals captured by the data acquisition boards). Notice the "mutations" in the genetic code of the solutions obtained by evolution (vertical chromosomes R24 to R1 reading from top to bottom, corresponding to switches S24 to S1 in Figure 1) compared with the human-designed circuit (rightmost vertical string).

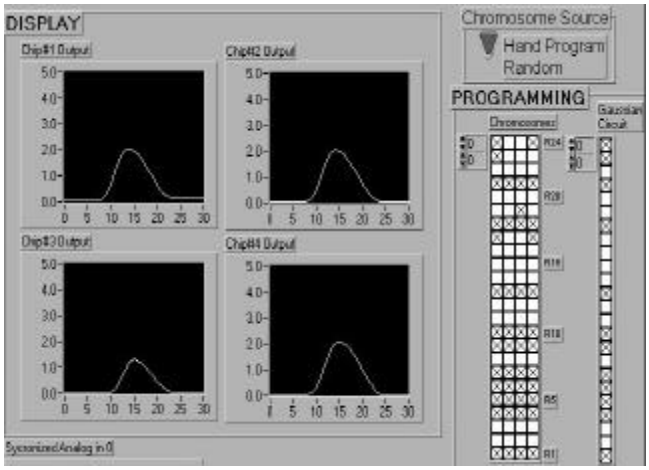

Figure 5 The "Gaussian" response of four "mutants" and their "genetic code" compared to the code of a human-designed circuit.

5. A Self-healing Experiment

The aim of this experiment was to test the reliability of a circuit design obtained by evolution and the availability of the electronic circuit using the on-line self-repairing property of the evolutionary mechanism [14]. Two FPTAs were cascaded interconnecting them by three external wires. The connection terminals P2-Drain, P4-Drain and N6-Source of the first FPTA were connected respectively to P3-Source, N5-Drain and N7-Drain. The input voltage was injected to the N6-Gate of the first FPTA and the output load was connected to the P4-Source of the second FPTA. Both FPTAs received a current bias at the N7-Drain terminal.

Evolution started with a randomly initiated population of coded configurations, which were transformed into connection patterns; these were downloaded to the chip. The output of the generated circuits was compared with the desired DC Gaussian and their difference was transformed in a fitness function (which should in the ideal case be zero or very small). During the evolution the fitness function shows improvements of the search as illustrated in figure 6. The codes for circuits generating best responses (i.e. closest to target according to some metric) were selected,

and suffered genetic operations, as controlled by the evolutionary algorithm. After looping for a number of times (75 generations), a circuit that best satisfied the requirements was found and left operational to provide the desired function. The performance of the chip continued to be monitored using the fitness function.

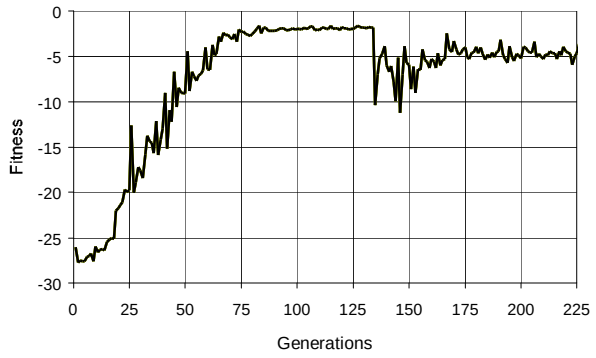

Figure 6 Fitness value monitoring the performance of the circuit. At generation 134, we inject a fault by removing one external wire between the two FPTA's.

At any time if the performance decreases below a certain threshold (e.g. when a fault is injected), the evolution process restarts the search for a new circuit configuration, taking into account the previous circuit configurations in the population. In this experiment, a fault was injected by disconnecting one of the external connection between the two FPTAs used by the operational circuit. At that time a lowering of performance but not a complete failure was observed. The reason for the graceful degradation is that the population of circuits obtained by the evolution process contains mutants insensitive to faults having the same phenotypic effect as a genetic mutation as shown on figure 7.

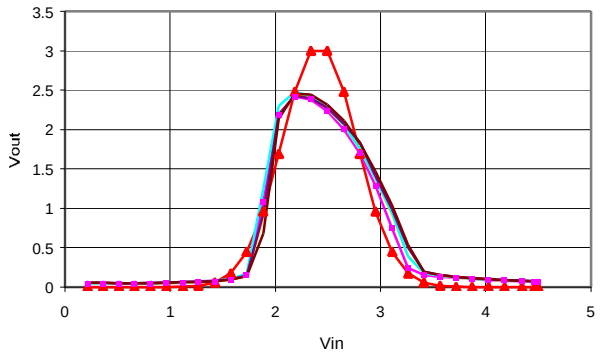

Figure 7 Target Gaussian response (triangle marker), the best individual (square marker) and its four mutants insensitive to faults having the same phenotypic effect as a genetic mutation.

When the fault was injected the GA restarted with the population of its last run, which included the solution that was currently affected by fault and some of its mutants. The faulty part became just another component to be used: the evolutionary algorithm did not "know" that the part was supposed to do something else. While starting with a random population took about the same time as finding a solution in the first place (not shown), starting with the last available population led to recovery in about 1/3 of the time while the circuit performance recovered to 90% (shown in figure 8).

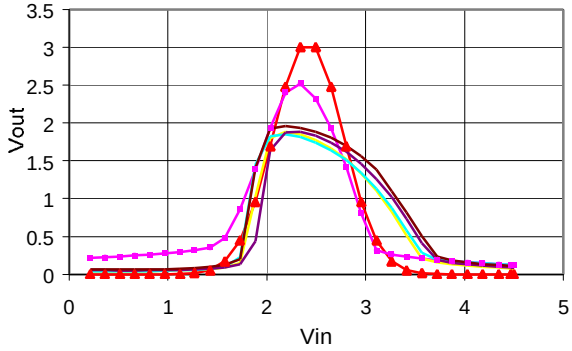

Figure 8 Target Gaussian response (triangle marker), the best individual (square marker) and its four mutants after on-line self-repaired by evolution.

The experiments used a Genetic Algorithm (GA) with the following parameters: population 500, mutation probability: 0.04, cross-over probability: 0.7, elite strategy: 10%, fitness function: mean square error. The GA obtained the Gaussian response before 100 generations, in about 14 minutes and recovered the fault in about 4 minutes.

6. Lessons Learned

Speed-up by evaluations in hardware

Hardware evaluation can produce a speed-up, especially when one simulates large, complex analog circuits, and the circuit response is rapid. One aspect that can however be easily overlooked is the frequency of operation for which a certain circuit is designed. There are limitations to increasing the speed of configuration and test in hardware. For example, the output of the Gaussian circuit on the FPTA started attenuation when the input ramp signals were exceeding 1kHz. Thus, no more than 1000 circuits per second (of desired low frequency response) could be reliably evaluated. Even though some artifacts of the particular FPTA design and load choice may be involved, it appears natural that evaluating the circuits at a different frequency than that of intended functioning may introduce errors. Evaluation in parallel is an alternative speed-up technique, and at least in the experiments with the FPTA chips no significant differences were noted between the implementation of the same circuit on different chips

Effect of Evolution for Fault-tolerance and self-healing

Some insensitivity to faults that has the same influence on the circuit as a genetic mutation tends to arise for free when using evolution. Tolerance to an arbitrary and large set of faults can possibly be achieved by testing the individuals circuit in the presence of possible faults, athough it may be time-consuming. We observed also that defects that are permanent have properties that are put to use for on-line self-repair. It would be interesting to evaluate the combination of the evolutionary approach and the more traditional redundancy methods such as explored in the "embryological" development approach [13]. These initial experiments while illustrating the power of evolutionary algorithms to design digital and analog circuit and to maintain functionality by recovering from faults without explicit redundancy, only prepare the ground for further questions. Examples of further questions include addressing how can the evolutionary mechanism be protected such that its implementation is not itself subject to faults, or how should the fitness function be computed/stored.

8. CONCLUSION

This paper demonstrates two features enabled by evolvable hardware and which may play an important role in flexibility and survivability of future space hardware. These features are automatic synthesis of circuits to perform new functions and self-healing – recovery from faults.

ACKNOWLEDGEMENTS

The research described in this paper was performed at the Center for integrated Space Microsystems, Jet Propulsion Laboratory, California Institute of Technology and was sponsored by the National Aeronautics and Space Administration.

REFERENCES

[1] J. Koza, F.H. Bennett, D. Andre, and M.A Keane, "Automated WYWIWYG design of both the topology and component values of analog electrical circuits using genetic programming", *Proceedings of Genetic Programming Conference*, Stanford, CA , pp. 28-31, 1996

[2] J. Lohn, J. and S. Colombano, "Automated Analog Circuit Synthesis using a linear representation", M. Sipper, D. Mange and A. Perez-Uribe (Eds.*) Evolvable Systems: From Biology to Hardware*, Springer-Verlag Lecture Notes in Computer Science Berlin 1998, pp. 125-133

[3] A. Thompson, "An evolved circuit, intrinsic in silicon, entwined in physics". In *International Conference on Evolvable Systems*. Springer-Verlag Lecture Notes in Computer Science, 1996, pp. 390-405.

[4] E. Sanchez and M. Tomassini (Eds.) *Towards Evolvable Hardware*, LNCS 1062, Springer-Verlag, 1996

[5] T. Higuchi, M. Iwata, and W. Liu (Eds.) Evolvable Systems: From Biology To Hardware, *Proc. of the First International Conference, ICES 96*, Tsukuba, Japan, Springer-Verlag Lecture Notes in Computer Science, 1997.

[6] M. Sipper, D. Mange, A. Perez-Uribe (Eds.) Evolvable Systems: From Biology To Hardware, *Proc. of the Second International Conference, ICES 98*, Lausanne, Switzerland, Springer-Verlag Lecture Notes in Computer Science, 1998.

[7] J. R. Koza, F. H. Bennett III,, D. Andre and M. A. Keane, Genetic Programming III – Darwinian Invention and Problem Solving, Morgan Kaufman, San Francisco, 1999

[8] E. Vitoz, Analog VLSI Processing: Why, Where and How, *Journal of VLSI Processing*, Kluwer, 1993

[9] Stoica, A. On hardware evolvability and levels of granularity. *Proc. of the International Conference "Intelligent Systems and Semiotics 97: A Learning Perspective*, NIST, Gaithersburg, MD, Sept. 22-25, 1997

[10] Stoica, A. Klimeck, G. Salazar-Lazaro, C. Keymeulen, D. and Thakoor, A. Evolutionary design of electronic devices and circuits, *Proc. of the 1999 Congress on Evolutionary Computation*, Washington, DC, July 6-9, 1999

[11] Stoica, A. Toward evolvable hardware chips: experiments with a programmable transistor array. *Proceedings of 7th International Conference on Microelectronics for Neural, Fuzzy and Bio-Inspired Systems*, Granada, Spain, April 7-9, IEEE Comp Sci. Press, 1999.

[12] Layzell, P. A New Research tool for Intrinsic Hardware Evolution , *ICES 98*. Springer-Verlag Lecture Notes in Computer Science, 1998

[13] P. Marchal et al. Embryological development on silicon. In R. Brooks and P. Maes, editors, *Artificial Life IV*, pages 365-366. MIT Press, 1994.

[14] White R. and Miles F. Principles of Fault Tolerance. In *Proceedings of Eleventh Annual Applied Power electronic Conference and Exposition*, pages 18-25, Vol.1. IEEE Press, 1996.

[15] Thompson A. In *Proceeding of the First Interntional Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, pages 524-529. IEEE Press, 1995.

[16] Layzell, P. In *Proceedings of the First NASA/DoD Worshop on Evolvable Hardware*, pages 85-86. IEEE Computer Society Press, 1999.

[17] Stoica A., Keymeulen D., Tawel R., Salazar-Lazaro C., Li W. In *Proceedings of the First NASA/DoD Worshop on Evolvable Hardware*, pages 76-84. IEEE Computer Society Press, 1999.

[18] Zebulum, R. et al., "Analog Circuits Evolution in Extrinsic and Intrinsic Modes" In *Proc. of the Second International*

Conference, ICES 98, pages 154-165. Springer-Verlag Lecture Notes in Computer Science, 1998.

Adrian Stoica is a Senior Member of Technical Staff at Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA.His research interests include learning and adaptive hardware, evolvable hardware, sensor fusion processors, robot learning, and humanoid robots. He has published more than 50 papers in these areas. He has a Ph.D. in EE from Victoria University of Technology, Melbourne, Australia, and a MSEE from Technical University of Iasi, Romania.

Didier Keymeulen *is a Research Engineer at the Jet Propulsion Laboratory of the California Institute of Technology.. His interests are in complex dynamical systems applied to the design of adaptive embedded systems. He obtained his MS. and Ph.D. in Electrical Engineering and Computer Science from the Artificial Intelligence Laboratory of the Vrije Universiteit Brussel, Belgium.*

Carlos Harold Salazar-Lazaro *is a PhD Student in Mathematics at Caltech, Pasadena, CA. He received his BS and MS in Computer Science and Mathematics from Rensselaer Polytechnic Institute. During his internship at JPL he worked on evolutionary synthesis of electronic circuits. He developed the evolutionary synthesis software for the HP-Exemplar supercomputer, and performed simulated evolutions of analog circuits.*

Vu Duong *is a Master Student at U.C. Irvine. He got his BS in Computer Science from UCSD. During his internship at JPL he programmed the evolutionary hardware test bed using LabView and integrated measurement instruments.*